WE CLAIM:

1. A method of operating a pipelined digital processor having a memory, comprising:

defining a first instruction, said first instruction being adapted to stall the pipeline of said processor upon execution thereof;

providing said first instruction within said pipeline;

decoding said first instruction;

executing said first instruction;

stalling said pipeline in response to said first instruction;

disabling said memory in response to said first instruction; and

restarting said pipeline and enabling said memory upon the occurrence of a predetermined event.

2. The method of Claim 1, wherein said predetermined event comprises a program interrupt.

3. The method of Claim 2, wherein said program interrupt comprises transfer of programmatic control to an interrupt service routine.

4. The method of Claim 1, wherein said predetermined event comprises a restart condition, said processor being re-enabled after having been halted.

5. The method of Claim 1, further comprising waiting for a wait state duration time after said act of disabling but before said pipeline is restarted.

6. The method of Claim 2, further comprising preventing the setting of interrupt flags from that point when the interrupt flags are cleared until said pipeline is stalled.

7. The method of Claim 1, wherein said act of providing said first instruction within said pipeline comprises:

providing a flag setting branch instruction having a delay slot within said pipeline;

disposing said first instruction in said delay slot of said flag setting branch instruction.

8. The method of Claim 1, further comprising:

providing a logic circuit adapted for detection of a predetermined condition of the data within the pipeline;

inserting data into the pipeline;

detecting, using said logic circuit, that the predetermined condition exists with respect to certain of the data;

invoking a sleep mode within the pipeline in response to said detected condition if no such sleep mode is already invoked; and

permissively restarting the pipeline when the condition no longer exists.

9. The method of Claim 8, wherein said act of permissively restarting said pipeline comprises restarting said pipeline only if restart has been or is concurrently enabled by the occurrence of said predetermined event.

10. The method of Claim 8, wherein said act of detecting said predetermined condition of said data comprises using said logic circuit to detect when said data will not be used in a later stage of said pipeline.

11. The method of Claim 10, wherein said act of detecting when said data will not be used comprises detecting the activation of first and second enable signals, said first and second enable signals being activated if the current pipeline stage contains valid data.

12. The method of Claim 11, further comprising:

providing a plurality of extension instructions within the instruction set architecture of said processor;

wherein said act of activating said second enable signal comprises enabling the data path to the arithmetic logic unit (ALU) with respect to all of said plurality of extension instructions.

13. The method of Claim 8, wherein said act of detecting said predetermined condition comprises detecting the anticipatory execution of an instruction within said pipeline, said instruction being subsequently stopped by a conditional evaluation conducted by said processor.

14. The method of Claim 10, wherein said act of detecting said predetermined condition comprises detecting the anticipatory execution of an instruction within said pipeline, said instruction being subsequently stopped by a conditional evaluation.

15. The method of Claim 1, further comprising switching off a plurality of clocks within said processor in response to said act of stalling.

16. The method of Claim 15, further comprising preserving the clocks serving the interface module and timer of said processor in an active state.

17. The method of Claim 1, further comprising changing the status of at least one debug flag, said act of changing status thereby disabling at least one debug clock associated with said processor.

18. The method of Claim 1, further comprising limiting the number of nodes within at least a portion of the gate logic of said processor that toggle per clock cycle.

19. The method of Claim 18, further comprising limiting the number of bits in a binary sequence present within said data that change per clock cycle to a predetermined number.

20. The method of Claim 15, further comprising limiting the number of nodes within at least a portion of the gate logic of said processor that toggle per clock cycle.

21. A method of operating a pipelined digital processor having a logic circuit adapted for detection of a predetermined condition with respect to at least a portion of the data within said pipeline, comprising:

inserting a plurality of data into said pipeline;

detecting, using said logic circuit, that the predetermined condition exists with respect to certain of said data;

stalling said pipeline in response to said detected condition if no such pipeline stall is already invoked;

checking for the presence of said condition at least once thereafter; and

restarting the pipeline when said detected condition no longer exists.

22. The method of Claim 21, wherein said act of restarting said pipeline comprises permissively restarting said pipeline only if restart has been or is concurrently enabled by the occurrence of a predetermined event.

23. The method of Claim 22, wherein said predetermined event comprises a program interrupt request (IRQ).

24. The method of Claim 21, wherein said act of detecting said predetermined condition of said data comprises using said logic circuit to detect when said data will not be used in a later stage of said pipeline.

25. The method of Claim 24, wherein said act of detecting when said data will not be used comprises detecting the activation of first and second enable signals, said first and second enable signals being activated if the current pipeline stage contains valid data.

26. The method of Claim 25, wherein said act of activating said second enable signal comprises enabling the data path to the arithmetic logic unit (ALU) with respect to all extension instructions of said processor.

27. The method of Claim 21, wherein said act of detecting said predetermined condition comprises detecting the anticipatory execution of an instruction within said pipeline, said instruction being subsequently stopped by a conditional evaluation conducted by said processor.

28. A digital processor, comprising:

a pipeline having at least fetch, decode, and execute stages, said pipeline adapted to process a plurality of instructions and data therein, said pipeline further being adapted to allow for stalling thereof, said plurality of instructions comprising at least one extension instruction;

an arithmetic logic unit (ALU) operatively coupled to said pipeline, said ALU processing at least a portion of said data based at least in part on said at least one extension instruction; and

logic operatively coupled to said pipeline and adapted to:

(i) detect the validity of at least a portion of said data present in a first stage of said pipeline;

(ii) initiate a stall condition in said pipeline;

(iii) re-evaluate the validity of said data at least once after said stall condition is initiated; and

(iv) remove said stall condition when said at least portion of said data is valid.

29. The processor of Claim 28, further comprising first and second enable signal logic, said first and second enable signal logic generating respective first and second enable signals when said first pipeline stage contains valid data.

30. The method of Claim 29, wherein said second enable signal enables at least a portion of the data path to said ALU.

31. The processor of Claim 28, wherein said logic is further adapted to detect the anticipatory execution of a first instruction within said pipeline, said first instruction being subsequently stopped by a conditional evaluation conducted by said processor.

32. The processor of Claim 28, further comprising a plurality of clocks, wherein said processor is further adapted to switch off at least a portion of said plurality of clocks in response to said stall condition.

33. The processor of Claim 32, wherein said plurality of clocks excludes the clocks serving the interface module and timer of said processor.

34. The processor of Claim 28, further comprising:

at least one debug clock;

at least one debug flag; and

logic adapted to change the status of said at least one debug flag, said status change disabling said at least one debug clock.

35. The processor of Claim 28, wherein said logic is further configured to limit the number of bits present in a binary sequence that change per clock cycle to a predetermined number

36. A digital processor, comprising:

a processor core having a pipeline with at least fetch, decode, and execute stages, said pipeline adapted to process a plurality of instructions and data therein, including at least one first instruction adapted to stall said pipeline;

an arithmetic logic unit (ALU) operatively coupled to said pipeline, said ALU being adapted to process at least a portion of said data based on said instructions;

first logic operatively coupled to said pipeline and adapted to detect the presence of said at least one first instruction within said pipeline and stall said pipeline upon execution thereof;

second logic operatively coupled to said pipeline and adapted to restart said pipeline after stalling upon the occurrence of a predetermined event.

37. The digital processor of Claim 36, further comprising:

a data storage device operatively coupled to said processor core; and

third logic operatively coupled to said first logic and said data storage device, said third logic being configured to disable said data storage device upon the stalling of said pipeline by said first logic.

38. The digital processor of Claim 37, further comprising fourth logic adapted to re-enable said data storage device upon restart of said pipeline by said second logic.

39. The digital processor of Claim 36, further comprising logic adapted to:

(i) detect the validity of at least a portion of said data present in said pipeline;

(ii) initiate a stall condition in said pipeline if said at least portion is not valid;

(iii) re-evaluate the validity of said data at least once after said stall condition is initiated; and

(iv) remove said stall condition when said at least portion of said data is valid.

40. The digital processor of Claim 39, further comprising first and second enable signal logic, said first and second enable signal logic generating respective first and second enable signals when said pipeline contains valid data.

41. The digital processor of Claim 40, wherein said second enable signal enables at least a portion of the data path to said ALU.

42. The digital processor of Claim 36, further comprising a plurality of clocks, wherein said processor is further adapted to switch off at least a portion of said plurality of clocks in response to said pipeline stall.

43. The digital processor of Claim 42, further comprising an interface module and timer, and wherein said plurality of clocks excludes the clocks serving said interface module and timer.

44. The digital processor of Claim 36, further comprising:

at least one debug clock;

at least one debug flag; and

flag setting logic adapted to change the status of said at least one debug flag, said status change disabling said at least one debug clock.

45. The digital processor of Claim 36, wherein at least a portion of said first or second logic is configured to limit the number of bits present in a binary sequence of said data that change per clock cycle to a predetermined number.

46. The digital processor of Claim 36, wherein said at least first instruction is disposed within a delay slot of a flag setting branch instruction.

47. The digital processor of Claim 46, wherein said branch instruction comprises a jump instruction, said jump being conditional on at least one condition within said pipeline.

48. The digital processor of Claim 39, wherein said plurality of instructions comprises at least one extension instruction, and said processor further comprises an extension ALU.

49. A method of operating a digital processor core having a multi-stage pipeline, a program counter (PC), a plurality of core registers, a storage device adapted to store a plurality of data therein, and a plurality of flags, including interrupt flags stored in said storage device, said processor core including an instruction set having at least one branch instruction and an associated delay slot, and at least one first instruction disposed in said delay slot and adapted to stall said pipeline upon execution, comprising:

storing the settings associated with said interrupt flags in a first of said core registers;

storing a destination address in said first core register;

temporarily blocking new interrupt requests;

processing all said interrupt flags stored in said storage device;

executing said branch instruction to branch to said first core register;

updating said PC with said destination address;

unblocking said interrupt requests; and

executing said first instruction to cause said pipeline to stall with no interrupt flags set in said storage device.

50. The method of Claim 49, wherein said act of blocking new interrupt requests comprises setting at least one interrupt enable flag.

51. The method of Claim 49, wherein said first instruction comprises a SLEEP instruction.

52. The method of Claim 49, wherein said at least one first instruction comprises a jump instruction.

53. The method of Claim 49, further comprising disabling at least a portion of said storage device in response to said execution of said first instruction.

54. The method of Claim 53, further comprising disabling at least one clock within said processor in response to said execution of said first instruction.

55. A digital processor, comprising:

a processor core having:

a multi-stage pipeline;

a program counter (PC);

a plurality of core registers;

a plurality of flags, including interrupt flags;

an instruction set having:

(i) at least one branch instruction and an associated delay slot; and

(ii) at least one first instruction disposed in said delay slot, said first instruction adapted to stall said pipeline upon execution: and

a storage device adapted to store a plurality of data therein, including said interrupt flags;

wherein said processor is adapted to stall said pipeline using the method comprising:

storing the settings associated with said interrupt flags in a first of said core registers;

storing a destination address in said first core register;

temporarily blocking new interrupt requests;

processing all said interrupt flags stored in said storage device;

executing said branch instruction to branch to said first core register;

updating said PC with said destination address;

unblocking said interrupt requests; and

executing said first instruction to cause said pipeline to stall with no interrupt flags set in said storage device.

56. The processor of Claim 55, further comprising logic operatively coupled to said pipeline and adapted to:

(i) detect the validity of at least a portion of data present in a first stage of said pipeline;

(ii) initiate a stall condition in said pipeline;

(iii) re-evaluate the validity of the data in said pipeline at least once after said stall condition is initiated; and

(iv) remove said stall condition when said at least portion of the data in said pipeline is valid.

57. The processor of Claim 55, further comprising:

at least one debug clock;

at least one debug flag; and

logic adapted to change the status of said at least one debug flag in response to said execution of said first instruction, said status change disabling said at least one debug clock.

58. The processor of Claim 55, further comprising apparatus adapted to disable at least a portion of said storage device in response to said execution of said first instruction.

59. A digital processor optimized for reduced power consumption, comprising:

a processor core having a multi-stage pipeline;

a storage device capable of storing a plurality of data therein;

a plurality of clock signal generators;

an instruction set having at least one first instruction, said first instruction being adapted to stall said pipeline upon execution thereof;

first logic adapted to disable at least a portion of said storage device in response to stalling of said pipeline by said at least one first instruction;

second logic adapted to secure at least a portion of said plurality of clock signal generators in response to said stalling of said pipeline.

60. The digital processor of Claim 59, further comprising third logic operatively coupled to said pipeline and adapted to:

(i) detect the validity of at least a portion of the data present in a first stage of said pipeline;

(ii) initiate a stall condition in said pipeline;

(iii) re-evaluate the validity of the data at least once after said stall condition is initiated; and

(iv) remove said stall condition when said at least portion of the data in said pipeline is valid.

61. The digital processor of Claim 59, wherein said instruction set comprises a branch instruction having a delay slot, said first instruction being disposed in said delay slot.

62. The digital processor of Claim 59, further comprising a timer, wherein said timer is adapted to generate an interrupt request upon the occurrence of a predetermined event, said interrupt request restarting said pipeline after stalling by said first instruction.

63. The digital processor of Claim 62, wherein said predetermined event comprises wrapping of the timer at its maximum value.

64. A method of operating a pipelined data processor having a program counter (PC), core registers, interrupt, and storage device, said processor further being configured with a sleep mode invoked using a sleep instruction, the method comprising:

storing at least one current flag setting in a first core register;

storing a destination address in said first core register;

disabling the interrupt enable in said core;

servicing any interrupt flags present in said storage device;

executing a jump instruction to said first register;

updating said PC with said destination address present in said first register;

enabling said interrupt enable in said core; and executing said sleep instruction to cause said processor to enter said sleep mode with said interrupt flags in said storage device cleared.

65. The processor of Claim 28, wherein said logic is further configured to limit the number of bits present in a binary sequence that change per clock cycle to a minimum number.

66. A digital processor, comprising:

a pipeline having at least fetch, decode, and execute stages, said pipeline adapted to process a plurality of instructions and data therein, said pipeline further being adapted to allow for stalling thereof, said plurality of instruction means comprising at least one extension instruction means;

means for performing arithmetic operations, said means being operatively coupled to said pipeline and processing at least a portion of said data based at least in part on said at least one extension instruction means; and

logic means operatively coupled to said pipeline and adapted to:

(i) detect the validity of at least a portion of said data present in a first stage of said pipeline;

(ii) initiate a stall condition in said pipeline;

(iii) re-evaluate the validity of said data at least once after said stall condition is initiated; and

(iv) remove said stall condition when said at least portion of said data is valid.

67. The processor of Claim 66, further comprising:

at least one debug clock means;

at least one means for flagging; and

logic means adapted to change the status of said at least one means for flagging, said status change disabling said at least one debug clock means.

68. A digital processor, comprising:

processor core means having a pipeline with at least fetch, decode, and execute stages, said pipeline adapted to process a plurality of instructions and data therein, including at least one first instruction adapted to stall said pipeline;

arithmetic logic means operatively coupled to said pipeline, said logic means being adapted to process at least a portion of said data based on said instructions;

means for detecting the presence of said at least one first instruction within said pipeline and stall said pipeline upon execution thereof;

means for restarting said pipeline after stalling upon the occurrence of a predetermined event.

69. A digital processor, comprising:

a processor core having:

a multi-stage pipeline means;

a program counter (PC) means;

a plurality of core register means;

a plurality of flags, including interrupt flags;

an instruction set having:

(iii) at least one branch instruction and an associated delay slot; and

(iv) at least one first instruction disposed in said delay slot, said first instruction adapted to stall said pipeline means upon execution: and

means for storing data adapted to store data, including said interrupt flags, therein;

wherein said processor is adapted to stall said pipeline means using the method comprising the steps of:

storing the settings associated with said interrupt flags in a first of said core register means;

storing a destination address in said first core register means;

temporarily blocking new interrupt requests;

processing all said interrupt flags stored in said means for storing;

executing said branch instruction to branch to said first core register means;

updating said PC means with said destination address;

unblocking said interrupt requests; and

executing said first instruction to cause said pipeline means to stall with no interrupt flags set in said means for storing.

70. A method of operating a pipelined digital processor having a memory, comprising the steps of:

defining a first instruction for stalling the pipeline of said processor upon execution thereof;

providing said first instruction within said pipeline for subsequent decoding;

decoding said first instruction to permit execution of said first instruction;

executing said first instruction;

stalling said pipeline in response to said execution of said first instruction;

disabling said memory in response to said first instruction to reduce power consumption; and

restarting said pipeline and enabling said memory upon the occurrence of a predetermined event.